ChatGPT was not used to create this presentation

Cloud Native

Next evolution of abstracting away infrastructure?

Guest Lecture
Cloud Computing WS23/24
Frankfurt University of Applied Sciences

17th January 2024

claranet® Make modern happen.

About

Fabian Dörk
Director Cloud Native Services
Claranet GmbH

- Diploma in **computer science** with focus on operating systems and networking technologies
- Studied **philosophy** with focus on meta & normative ethics
- **20 years of experience** in the IT industry
- Obsessed with **innovation, automation & tooling**
- **Leading an international team** of cloud-, devops-, kubernetes-, software engineers

About

Domenico Caruso
Team Lead – Cloud Native Engineering
Claranet GmbH

- Master in Theoretical and Computational Chemistry
- Over 10 years of experience in IT industry
- Leading and working together with a team of cloud-, kubernetes engineers

claranet® Make modern happen.

About the Team

- Multiple teams embracing software engineering, cloud based and native workload, linux & windows
- Several platforms: AWS, GCP, Azure and on-premise
- International: based in Germany, Spain, India with over 6 nationalities and languages
- Annual team and family event

claranet® Make modern happen.

About: Claranet group

At a glance

- Founded in 1996
- Owner- managed
- 600 Mio € annualised revenues
- More than 10.000 B2B customers
- Global reach with operations in 11 counties
- More than 3.500 employees

We are experts for modernizing and running critical applications, data and infrastructures 24/7

claranet® Make modern happen.

About: Claranet DACH

claranet®

AddOn | KHETO

PART OF CLARANET GROUP

AddOn: Experts for
SAP Services
Workplace & Collaboration
Trainings

Claranet: Experts for
Cloud Services
Container/Kubernetes
Cyber Security
Network Services

KHETO: Experts for
SAP Business Intelligence
SAP Business Warehouse
SAP Analytics

2000
Founded in Frankfurt

200+
Employees

5
Locations in DACH

2
Datacenter in Germany

Highly accredited with cloud vendors

Compliance

ISO 9001:2015

ISO 22301:2019

ISO 27001:2017 27017:2015 27018:2014

STAR SELF-ASSESSMENT

Trusted Cloud SERVICE 10005

SOC 2 - TYPE II ISAE 3402

CLOUD COMPUTING BSI C5 COMPLIANCE CRITERIA CATALOGUE

CHECK 28

Partnerships

Google Cloud Partner Managed Service Provider

Gold Microsoft Partner Microsoft | Azure Expert MSP

aws PARTNER Managed Service Provider

aws PARTNER Premier Tier Services

aws PARTNER Select Tier Training

SAP Silver Partner

Trainer Skills & Certified SAP Consultant SAP SAP Authorized Trainer 2021

Zertifizierungen und Partnerschaften beziehen sich u.U. nur auf einzelne Unternehmen der Claranet Gruppe

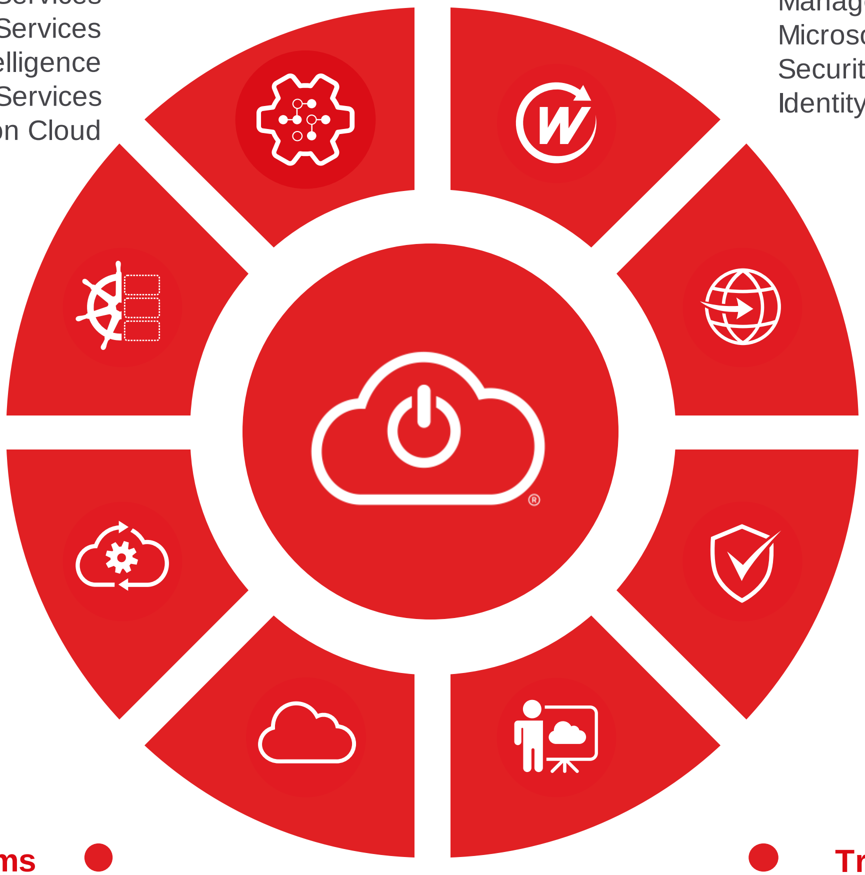
Claranet Service Portfolio

SAP
Managed Services
Consulting Services
Business Intelligence
Security Services
SAP on Cloud

Workplace & Collaboration
Managed Services
Microsoft 365
Security / Endpoint Management
Identity Management

Cloud Native
Managed Container Applications
Managed Kubernetes
DevOps Automation
Cloud Native Strategies

Network Services
Cloud Connect
MPLS
IPsec
SSL



Cloud Services
Cloud Migration
Cloud Management
Managed Applications

Cyber Security
Security Assessments
Penetration Testing & Red Teaming
Security & Compliance Consulting
SOC Services: EDR, MDR
Penetration Testing as a Service

Cloud Platforms
AWS, GCP, Azure
Private Cloud
Hybrid Cloud

Training
Microsoft
SAP
AWS
Linux
Cyber Security

24/7
Business SLA, Monitoring, Reporting, Service Management

What expects you in this lecture?

- Cloud is everywhere and it is well researched, well discussed, and well defined. Beyond the sheer fact of using someone elses computers new approaches emerging every day. One of them is the cloud native movement. This lecture gives a little bit of historic background where we are coming from and why something like cloud native exists in order to understand what the distinct traits compared to cloud based workload are.

- Aim of this lecture are
  - to make ourselves familiar with the **underlying concepts**
  - to realise what different kinds of **abstractions Kubernetes** introduces
  - to judge if the **introduced complexities** are for the good or for the bad

- Disclaimer: **It is an opiniated view on Cloud Native**, because we will focus on Kubernetes as the driver for cloud native approaches only. Serverless is for another occasion to be discussed.

Who heard of Kubernetes outside of this lecture?

Who gained real-world experience with Kubernetes?

Who thinks Kubernetes is a useful technology?

claranet® Make modern happen.

@theunsteady5

What is Cloud Native?

claranet® Make modern happen.

Premises

- **Software** generally considered as **competitive advantage** and therefore as essential mean for **value creation**
- Software development enters center stage
- **Agility** fosters **innovation** and **flexibility**
- **Microservices** architecture decouples subsystems which then could be developed, released, deployed independently
- **Parallelization** leads to increased **pace**
- **Infrastructure abstraction** enables developers to **focus** on **value creation**

Cloud-Native

claranet® Make modern happen.

Definition: Cloud Native

Cloud Native
*is structuring **teams**, **culture**, and **technology***
*to utilize **automation** and **architectures***
*to manage **complexity** and **unlock velocity.***

— Joe Beda (co-founder of Kubernetes)

claranet® Make modern happen.

Cloud-Native

</code>

Guiding principles

- Design for **performance** (responsiveness, concurrency, efficiency)
- Design for **automation** (of infrastructure and development tasks)
- Design for **resiliency** (fault-tolerance, self-healing)
- Design for **elasticity** (automatic scaling)
- Design for **delivery** (minimise cycle time, automate deployments)
- Design for **observability** (cluster-wide logs, traces, and metrics)

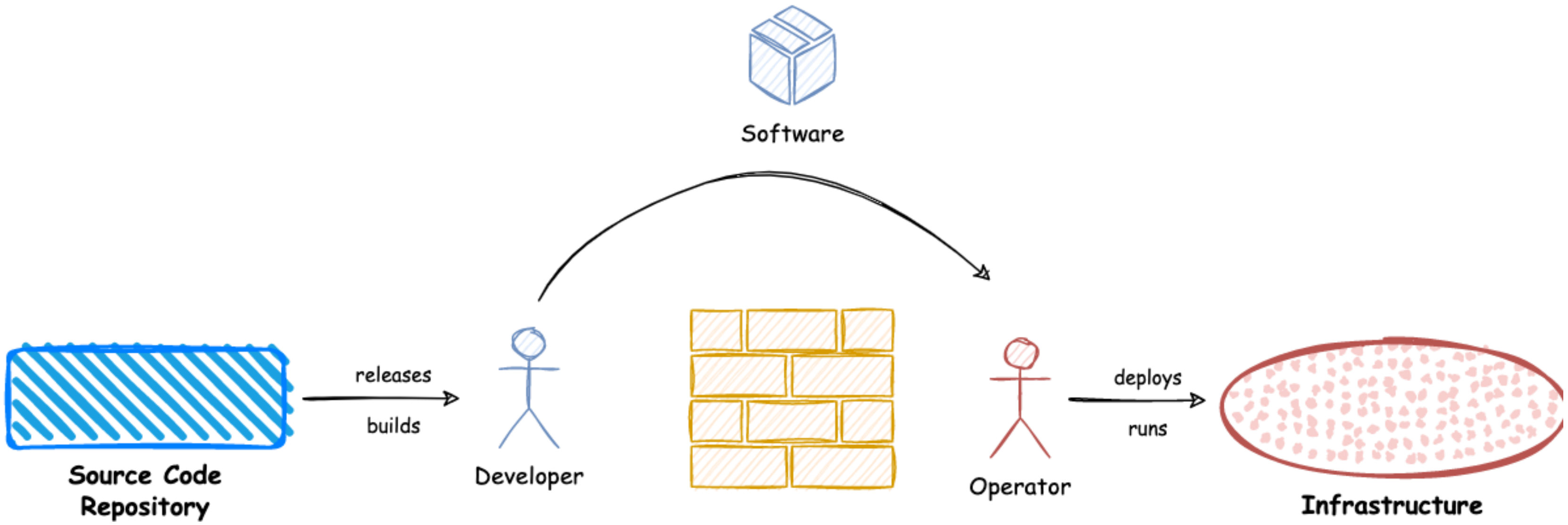
Cloud-Native


A bit of history
claranet
Make modern happen.

Silos, the old fashioned way

Risk loving software developer incentivised to introduce changes

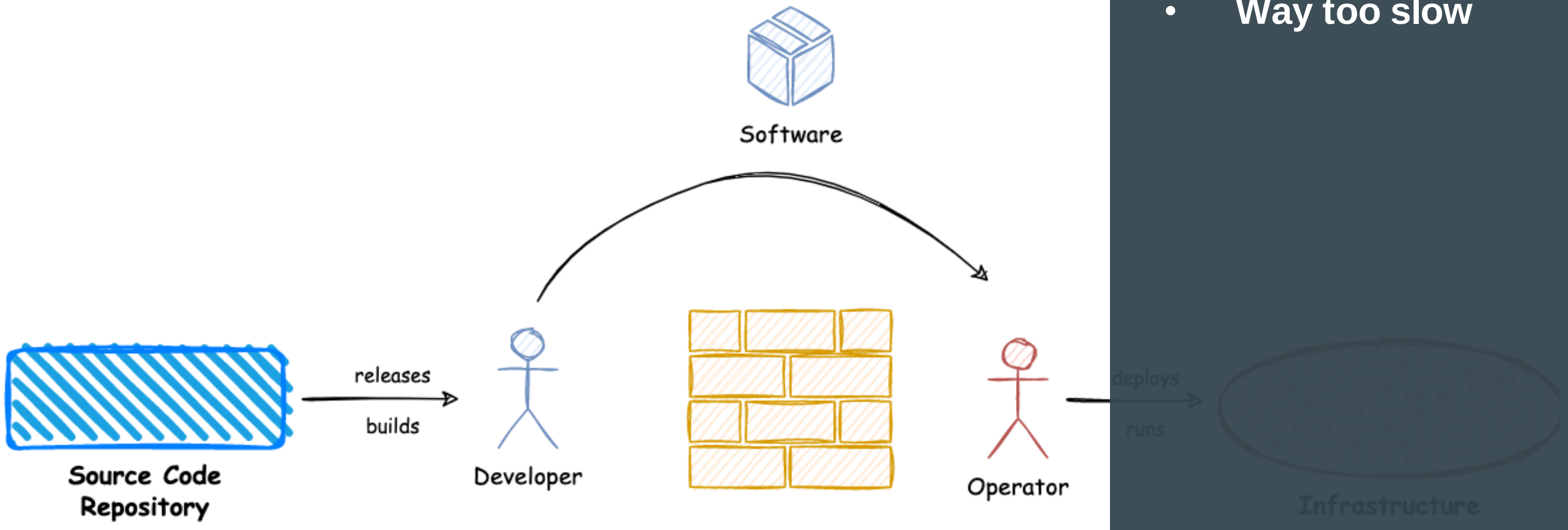
Risk averse IT operators incentivised to maintain stability



Silos, the old fashioned way

Risk loving software developer incentivised to introduce changes

Risk averse IT operators incentivised to maintain stability



claranet® Make modern happen.

Downsides:

- Conflicting aims
- Led to blame game: “It is your machines, not my code!”
- Way too slow

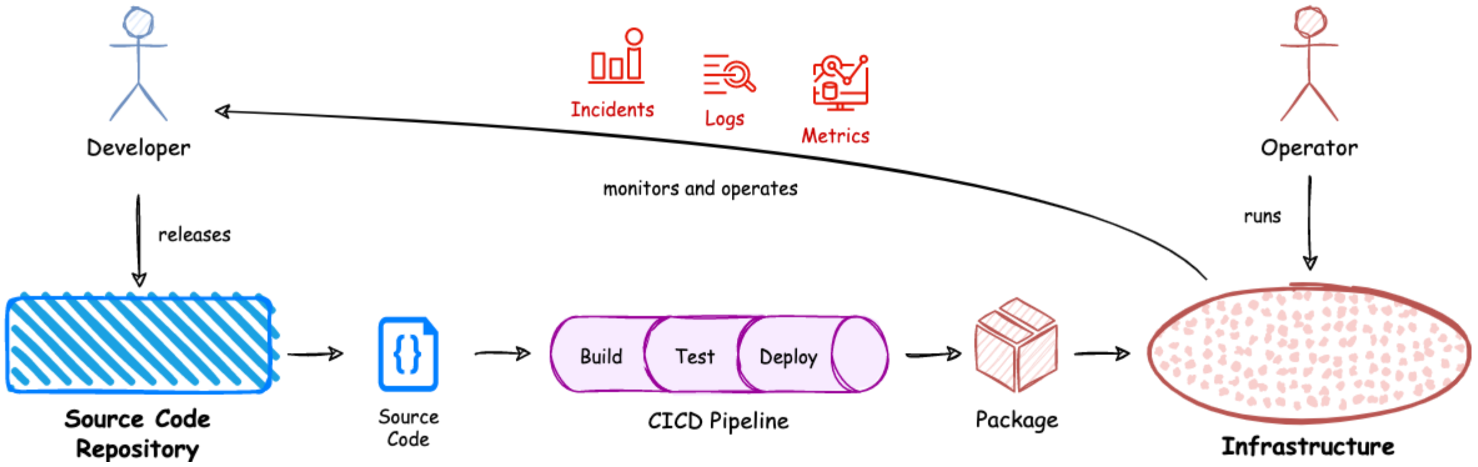
DevOps through Automation

Bridging the gap through tooling and automation

Brings together **skills, processes, and tools**

CICD pipelines ensures transparency, repeatability, and reliability

Feedback loop enables developers to react on incidents



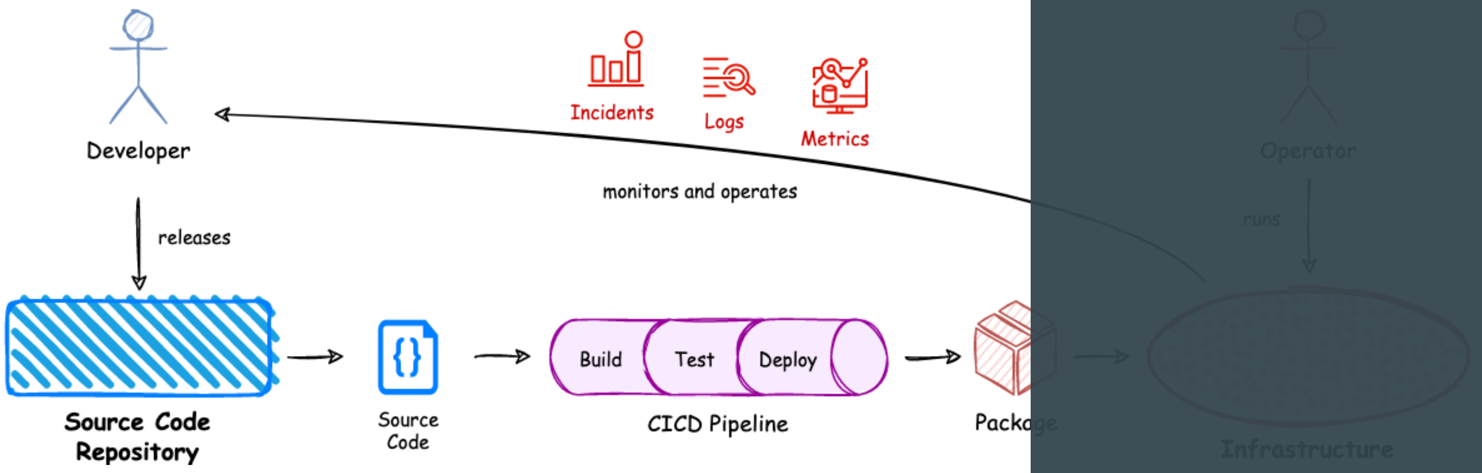
DevOps through Automation

Bridging the gap through tooling and automation

Brings together **skills, processes, and tools**

CICD pipelines ensures transparency, repeatability, and reliability

Feedback loop enables developers to react on incidents



Downsides:

- Demarcation line between Ops and Dev remains blurry
- It was still necessary for developers to SSH into servers in order to debug problems
- Still too slow

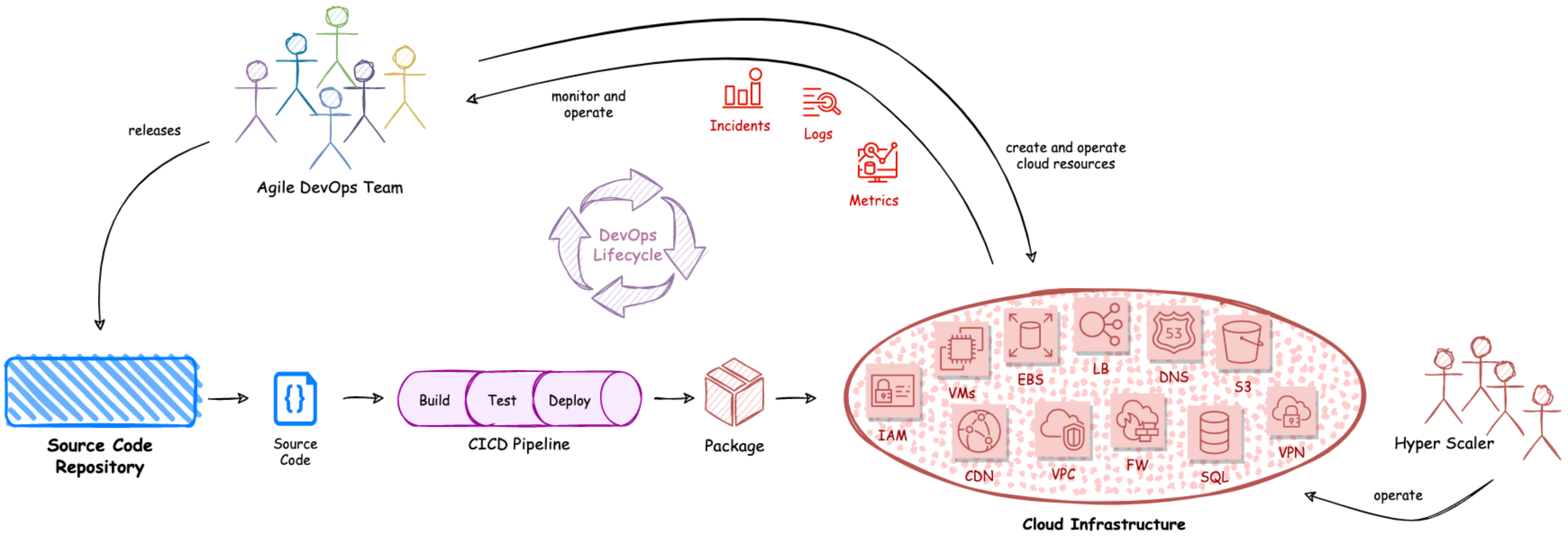
DevOps by Self Empowerment: You build it, you run it!

Cultural shift where teams embrace an **agile software engineering** approach, workflow, and toolset

Delivering business value by **tearing down silos**

Ownership of the entire stack belongs to a single **cross-functional team**

Continuous effort throughout the whole **software development lifecycle**



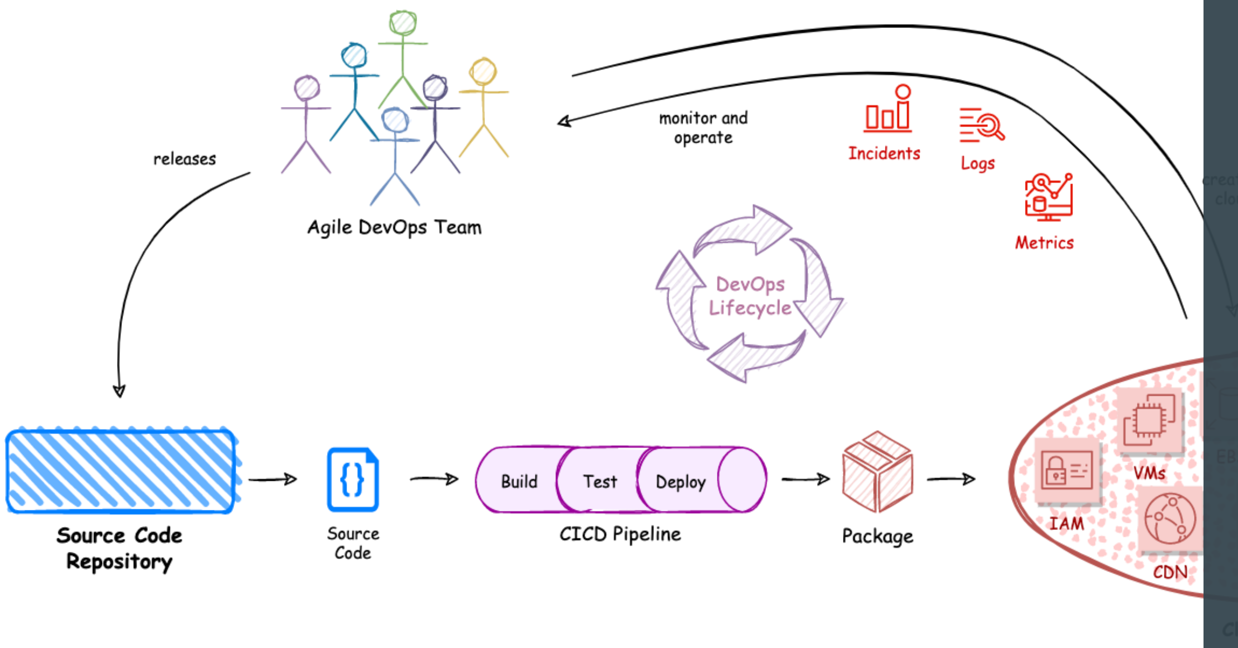
DevOps by Self Empowerment: You build it, you run it!

Cultural shift where teams embrace an **agile software engineering** approach, workflow, and toolset

Delivering business value by **tearing down silos**

Ownership of the entire stack belongs to a single **cross-functional team**

Continuous effort throughout the whole **software development lifecycle**



Downsides:

- Puts lots of burden onto developers
- Alignment across teams difficult without trading pace and flexibility
- Day 2 operations covered by developers
- Does not scale!

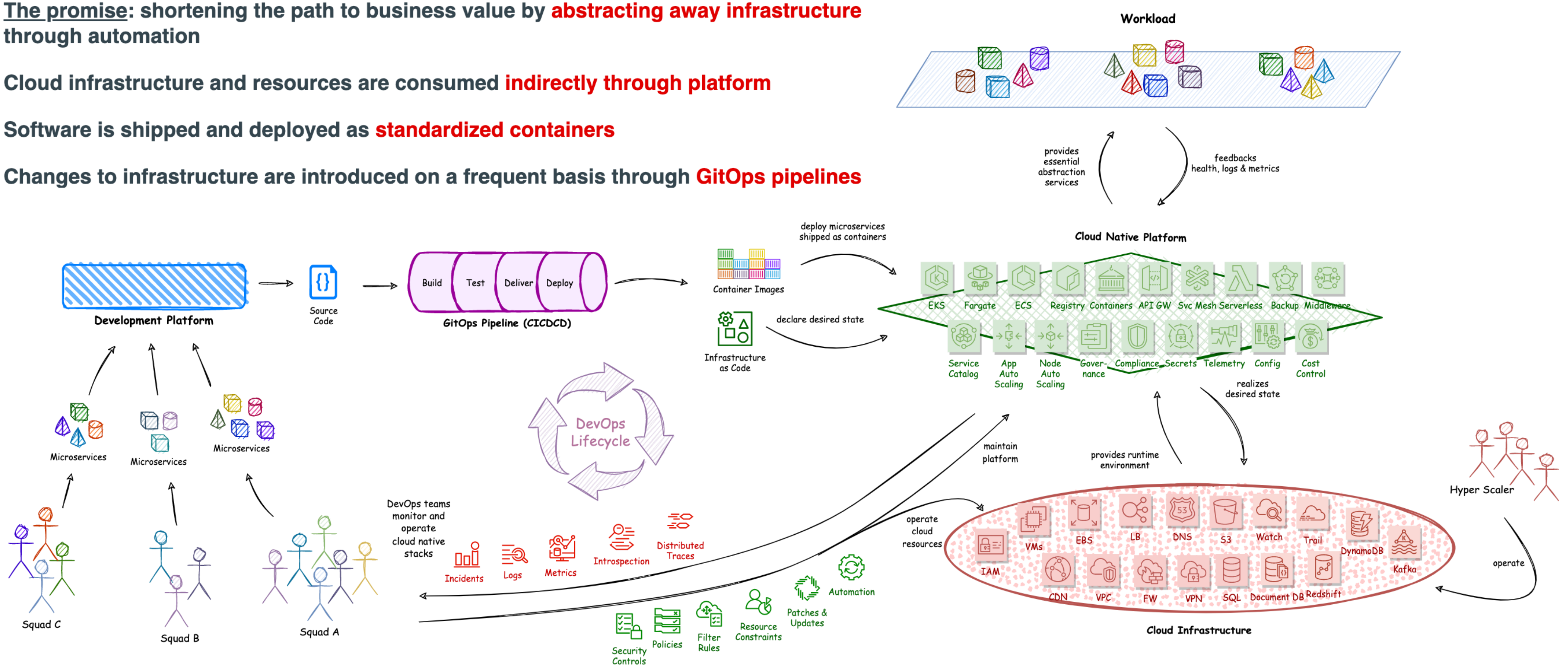
Cloud Native: Unlock the full potential of the cloud

The promise: shortening the path to business value by **abstracting away infrastructure** through automation

Cloud infrastructure and resources are consumed indirectly through platform

Software is shipped and deployed as standardized containers

Changes to infrastructure are introduced on a frequent basis through GitOps pipelines



claranet® Make modern happen.

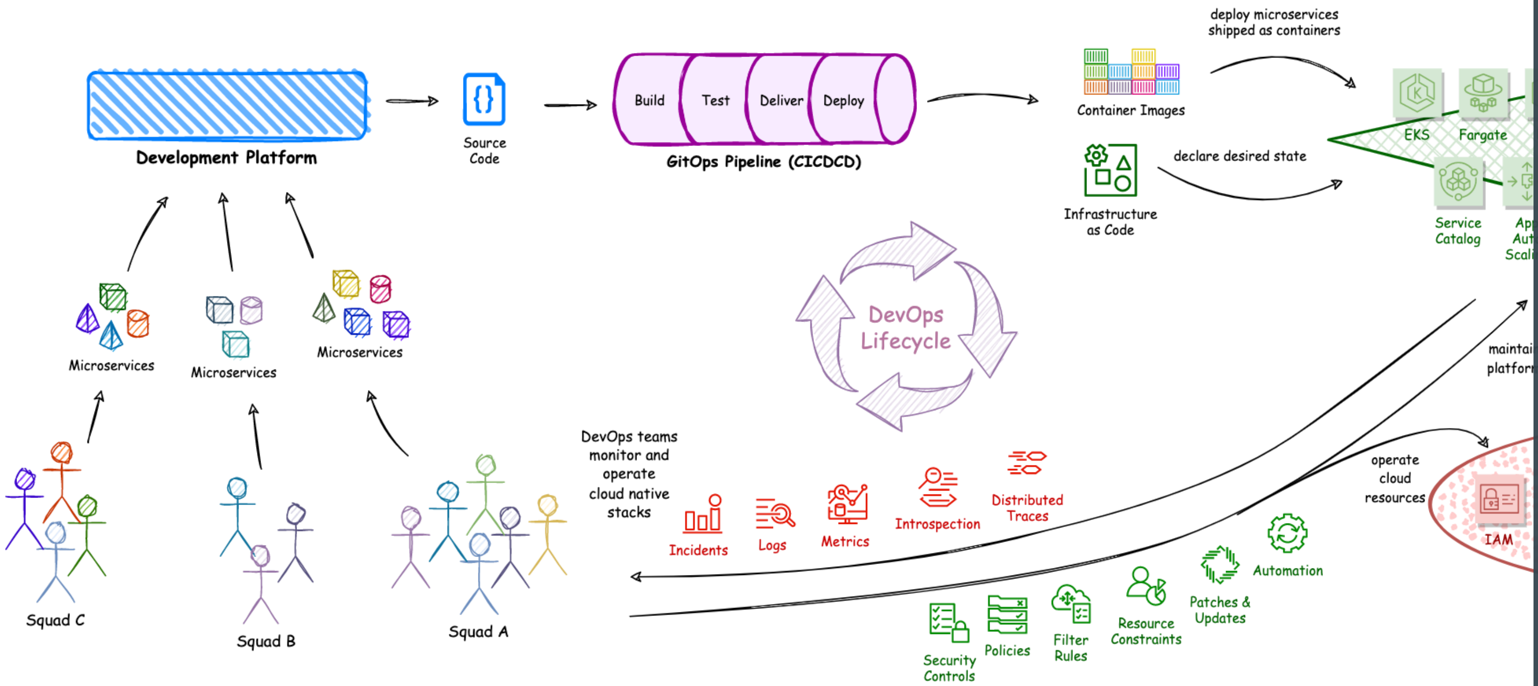
Cloud Native: Unlock the full potential of the cloud

The promise: shortening the path to business value by abstracting away infrastructure through automation

Cloud infrastructure and resources are consumed indirectly through platform

Software is shipped and deployed as standardized containers

Changes to infrastructure are introduced on a frequent basis through GitOps pipelines

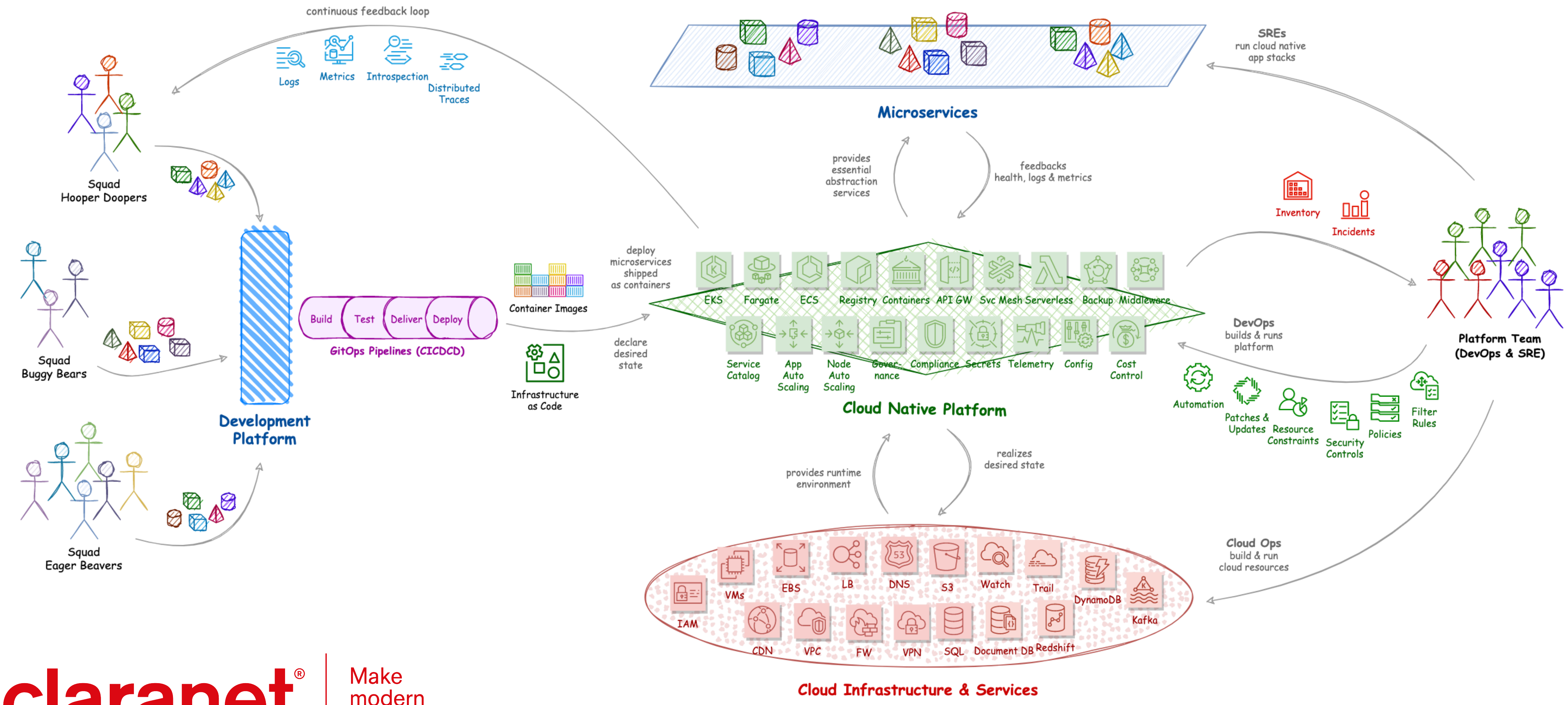


claranet® Make modern happen.

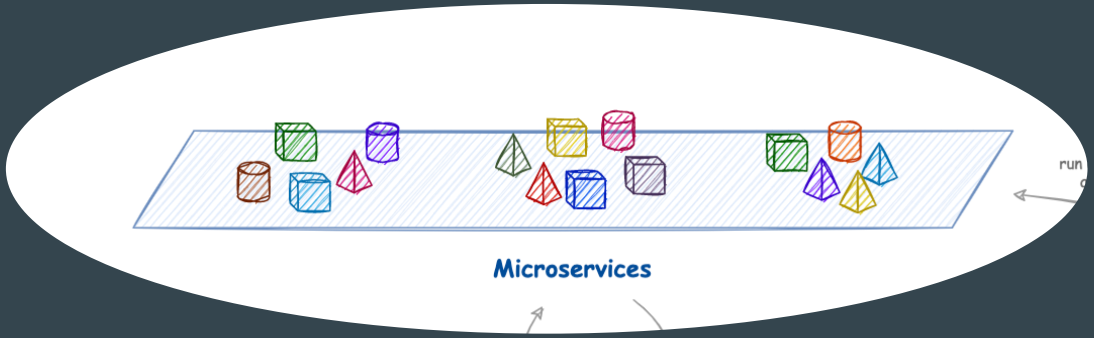
Downsides:

- Significantly greater complexity
- Day 2 operations covered by developers
- Focus on value creation jeopardized
- Who is responsible for compliance and security?

Cloud Native Reference Model



claranet® Make modern happen.

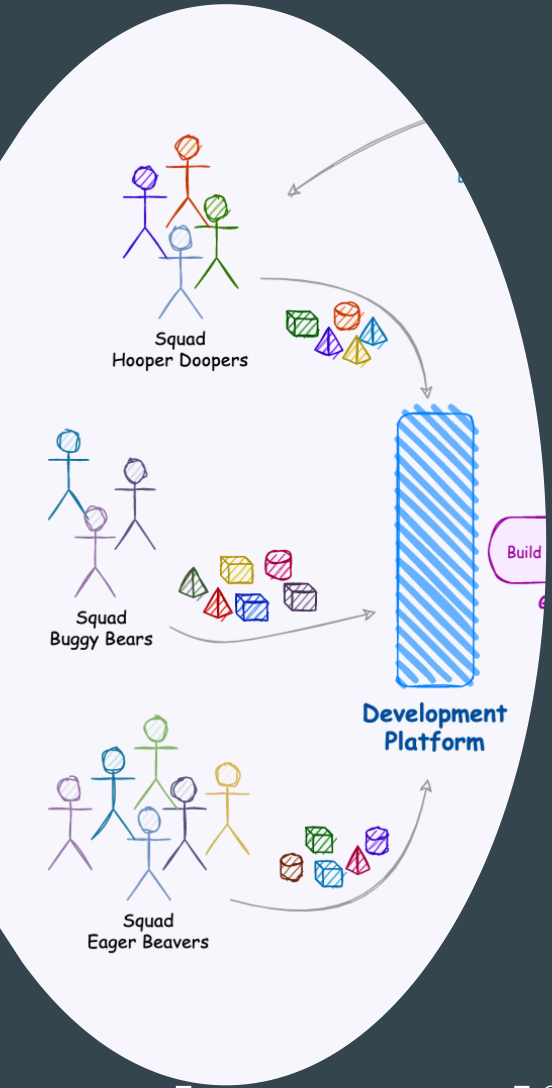


Application first, Infrastructure second!

Divide & Conquer: Decomposition of monoliths into microservices

Decoupling of software subsystems to minimise deadlocks and increase pace

Event-driven, asynchronous, scale-out technologies



Effectiveness over Efficiency!

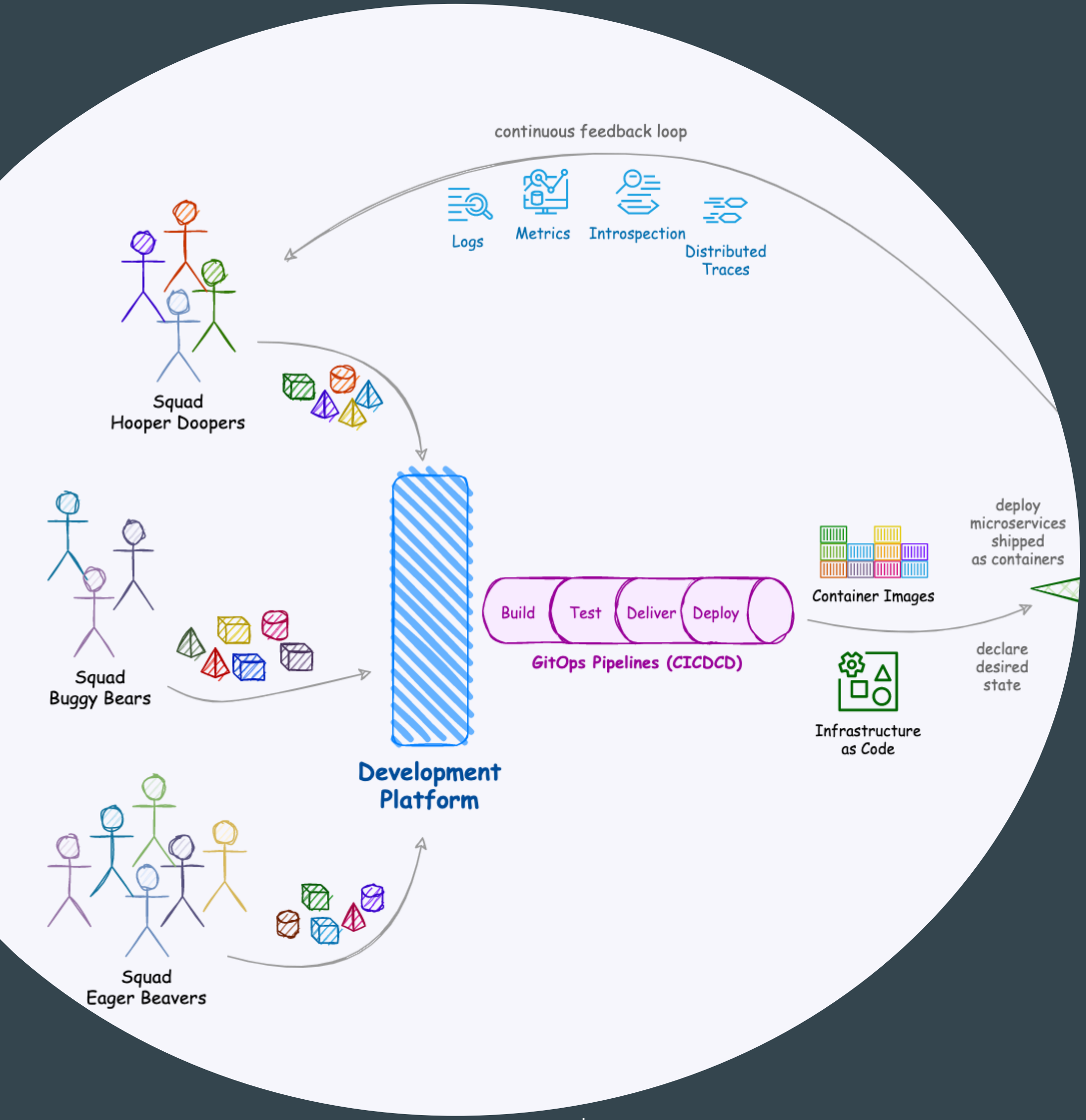
Agile teams develop, test, release and deploy software <u>independent</u> from each other

Agility promotes autonomy and <u>decentralized decision-making</u>

Gaining <u>pace</u> is the most important priority

Freeing developers from infrastructure and <u>day 2 operations</u>

claranet® Make modern happen.



Embrace GitOps!

Git is the <u>single source of truth</u>

Shipping software as <u>containers</u>

End to end automation from build, test, integration, delivery, deployment

<u>Autonomy</u> also in regard to infrastructure: software developers can easily deploy new services

Only way to introduce changes is through <u>pipelines</u> and well-defined <u>gates</u>

<u>Versioned infrastructure</u> expressed as declarative definitions of target state

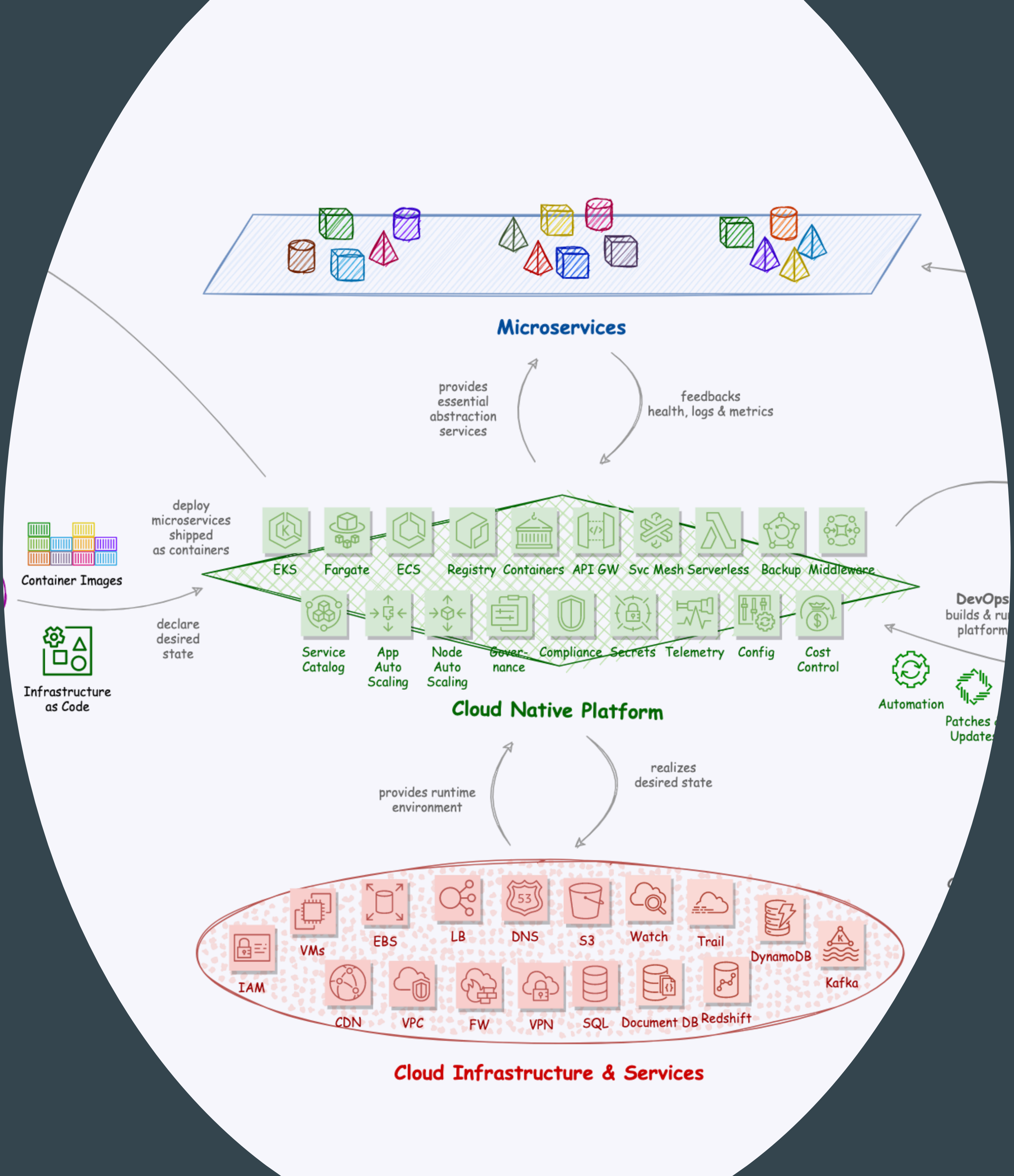
claranet® Make modern happen.

Decouple workload from infrastructure!

Abstraction layer provides translation & realisation

Kubernetes as a platform to build platforms

Platform sets boundaries, ensures alignment, enforces policies, checks compliance, and mitigates security risks



claranet® Make modern happen.

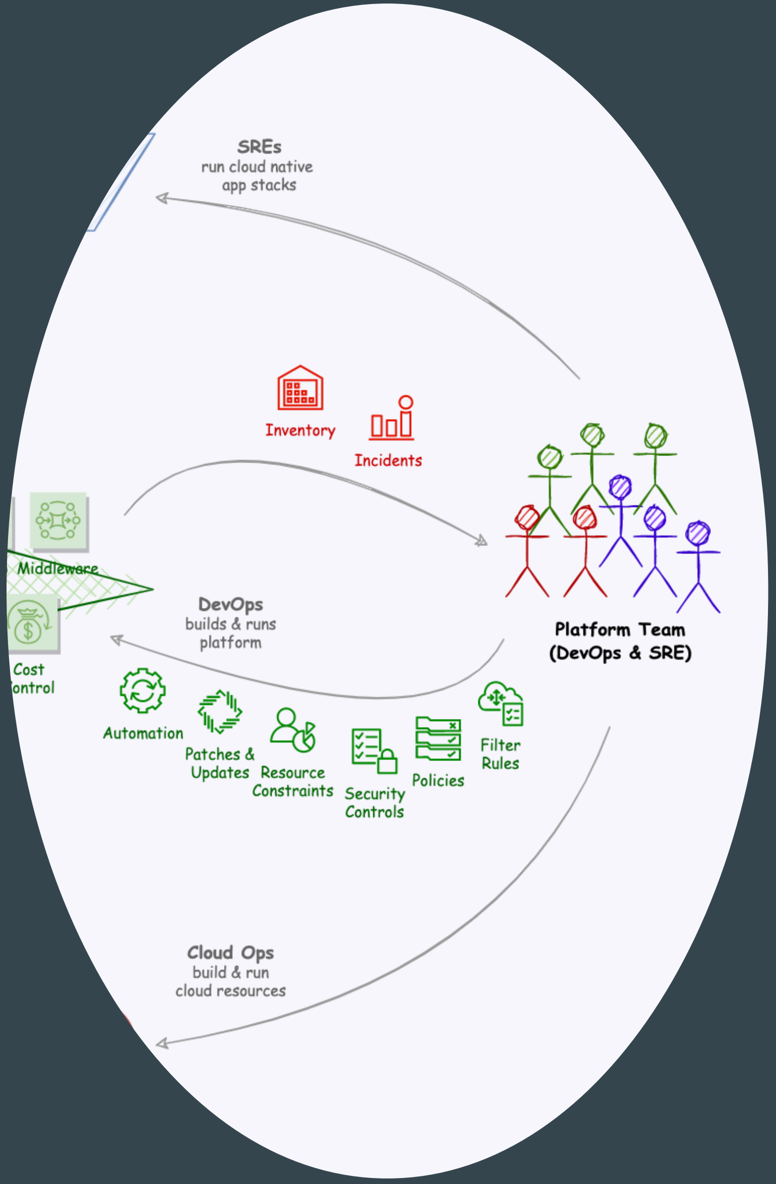
Tightly couple DevOps and DevOps!

<u>Specialised, cross-functional team</u> operates platform and microservices stack

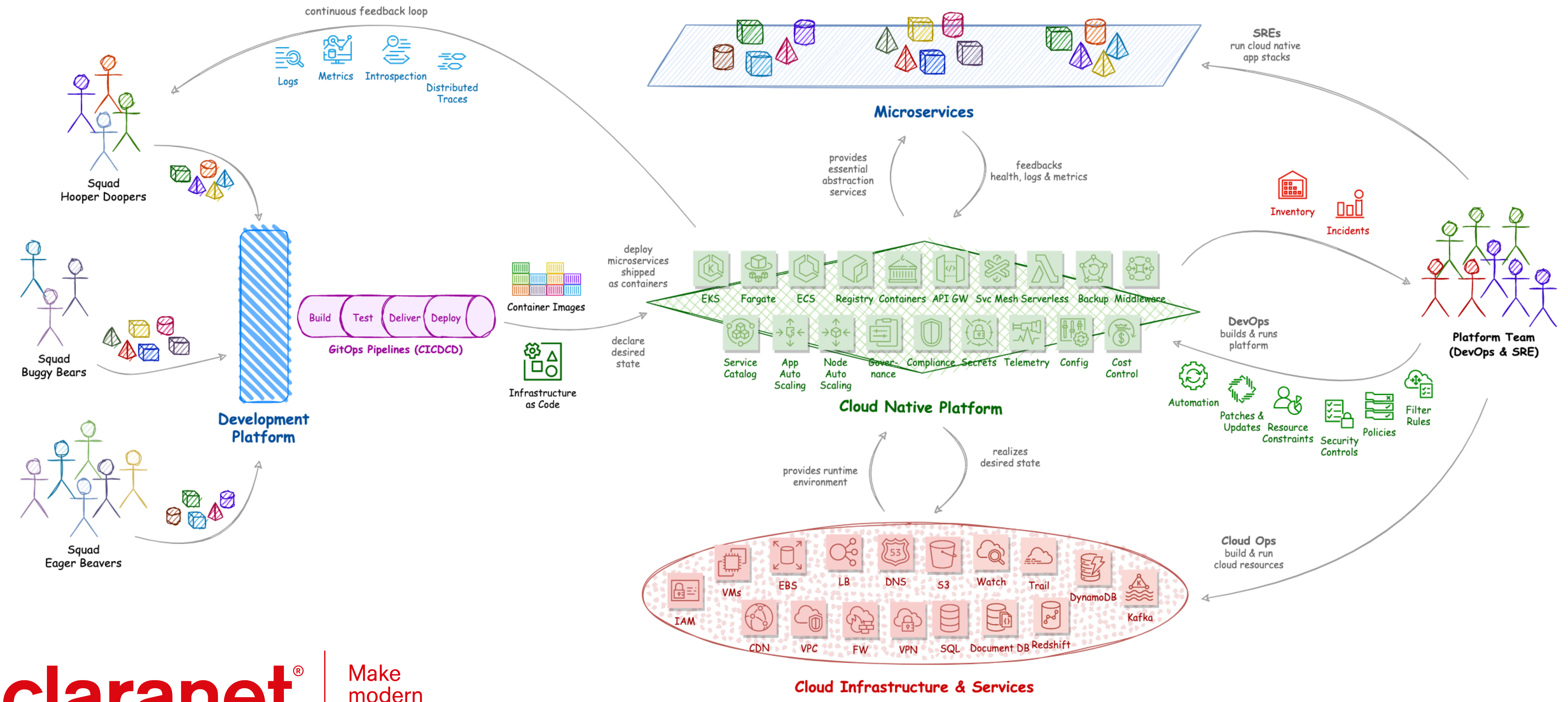
<u>Co-management</u> via shared responsibility model based on DevOps lifecycle

DevOps/SREs on both ends cooperate by sharing the same mindset

Platform itself is subject of the <u>software development lifecycle</u> and requires permanent advancements and modifications



Application first, infrastructure second!



claranet®

Make modern happen.

Cloud Native

Cloud
Flexible usage of infrastructure through automation to build highly scalable stacks

DevOps
Agile methodology rooted in the idea of cross-functional teams with shared responsibilities throughout lifecycle continuously producing value to the customer

Containers
Ship applications as standardised packages and run them as portable and lightweight operational unit with a well-defined interface

Microservices
Decomposition of monolithic applications into smaller building blocks to open up parallel development work streams

Kubernetes
Platform to deploy any set of applications shipped as containers transparently across a fleet of compute nodes and run them reliably and scalable

12 Factor App
Methodology for building SaaS apps that are suitable for deployment on modern cloud platforms

Continuous Delivery
Accelerate value delivery by leveraging automation to reliably push code into production continuously in a greater frequency

API First
A product centric approach to designing and developing consistent and reusable APIs which accelerates the development process, ensures interoperability, and fosters innovation

Infrastructure as Code
Managing and provisioning of cloud resources through declarative machine-readable definitions maintained in version control systems

Service Meshes
A tool for adding observability, security, and reliability features by transparently inserting this functionality at the platform layer

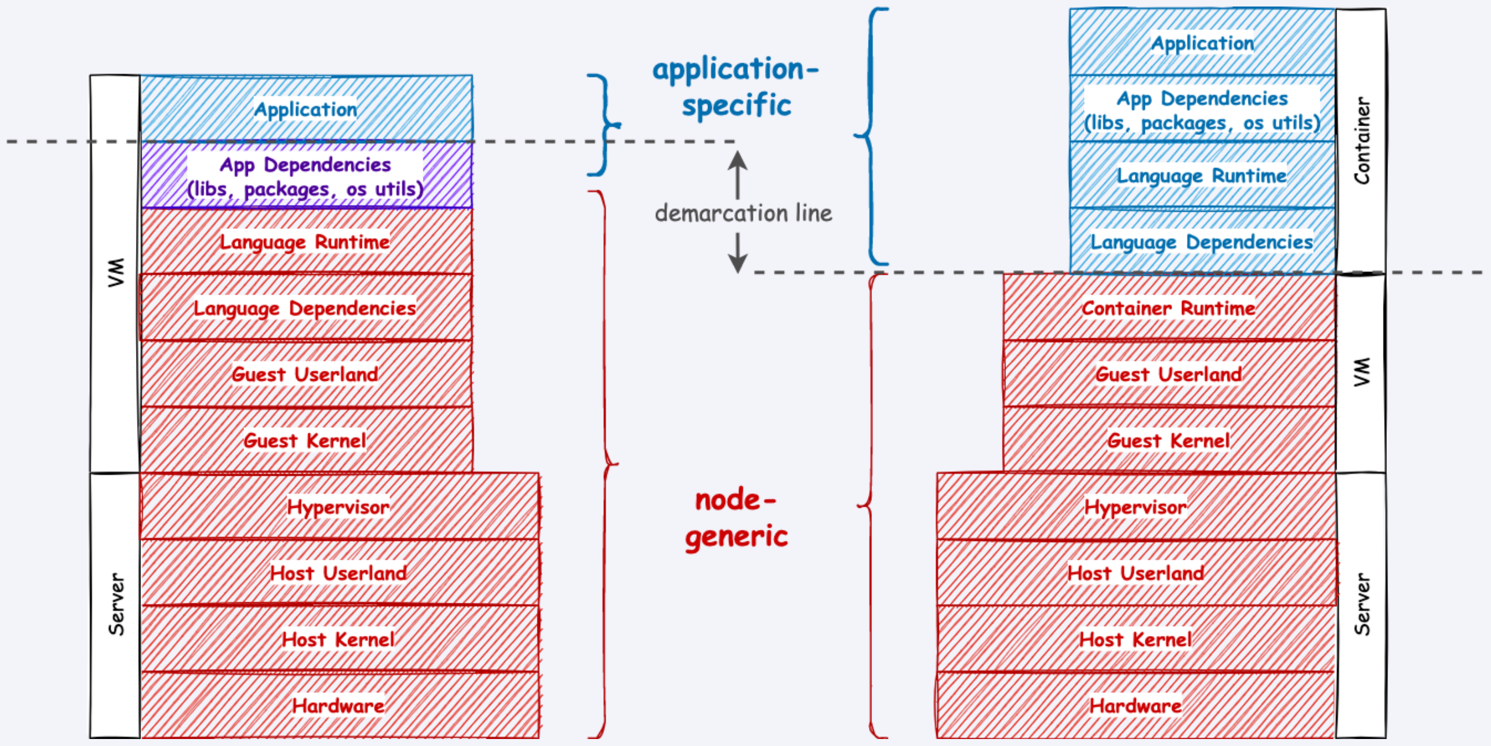
claranet®
Make modern happen.

Kubernetes architecture

Make modern happen.

Containers as universal shipping format

- **Lightweight os-level virtualisation**
- Isolation of resources (cpu, memory, storage, network)
- Initially based on Linux **cgroup v2**, **namespaces,** and union filesystems
- Standardized interface and format (OCI)
- Clear demarcation line between Dev and Ops
- **Tight coupling of application and runtime (!)**



So, what the heck is Kubernetes?

- ***„Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.“ **[*](#)***

- Trivia
  - Name originates from Greek, meaning helmsman or pilot ([ˌkuːbərˈnetiːz])
  - Abbreviated by k8s
  - Open-sourced in 2014 by Google
  - Hit the first production-grade version 1.0.1 in July 2015
  - Current version 1.27.1
  - Built upon 20 years of experience Google has with running production workloads at planet-scale
- CNCF is a vibrant community

- Papers
  - *„Large-scale cluster management at Google with Borg“*
    Abhishek Verma, Luis Pedrosa, Madhukar R. Korupolu, David Oppenheimer, Eric Tune, John Wilke; Proceedings of the European Conference on Computer Systems (EuroSys), ACM, Bordeaux, France (2015)
    https://research.google/pubs/pub43438/
  - *„Borg, Omega, and Kubernetes - Lessons learned from three container-management systems over a decade“*
    Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, John Wilkes; ACM Queue Volume 14, Issue 1, pp 70–93 (2016)
    https://dl.acm.org/doi/10.1145/2898442.2898444

Kubernetes Features

Automated rollouts and rollbacks

Kubernetes progressively rolls out changes to your application or its configuration, while monitoring application health to ensure it doesn't kill all your instances at the same time. If something goes wrong, Kubernetes will rollback the change for you. Take advantage of a growing ecosystem of deployment solutions.

Service discovery and load balancing

No need to modify your application to use an unfamiliar service discovery mechanism. Kubernetes gives Pods their own IP addresses and a single DNS name for a set of Pods, and can load-balance across them.

Storage orchestration

Automatically mount the storage system of your choice, whether from local storage, a public cloud provider such as AWS or GCP, or a network storage system such as NFS, iSCSI, Ceph, Cinder.

Self-healing

Restarts containers that fail, replaces and reschedules containers when nodes die, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.

Secret and configuration management

Deploy and update secrets and application configuration without rebuilding your image and without exposing secrets in your stack configuration.

Automatic bin packing

Automatically places containers based on their resource requirements and other constraints, while not sacrificing availability. Mix critical and best-effort workloads in order to drive up utilization and save even more resources.

Batch execution

In addition to services, Kubernetes can manage your batch and CI workloads, replacing containers that fail, if desired.

Horizontal scaling

Scale your application up and down with a simple command, with a UI, or automatically based on CPU usage.

IPv4/IPv6 dual-stack

Allocation of IPv4 and IPv6 addresses to Pods and Services

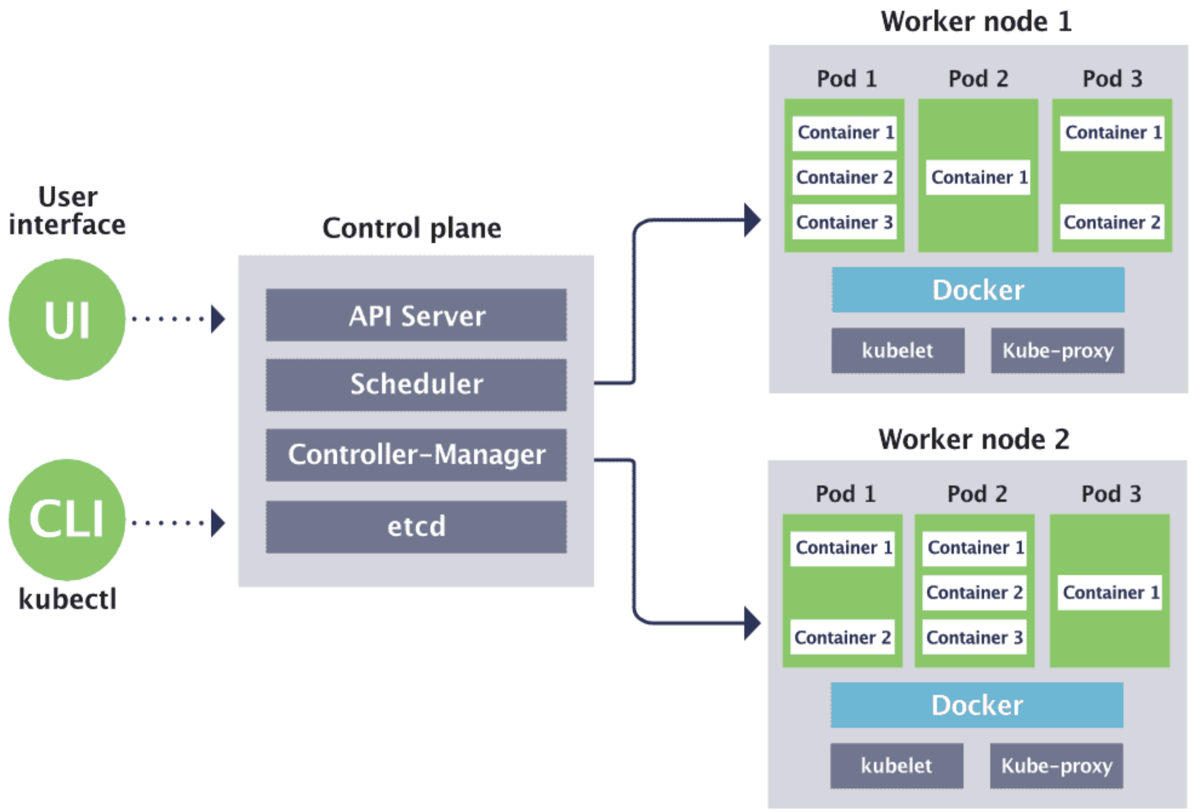
Designed for extensibility

Add features to your Kubernetes cluster without changing upstream source code.

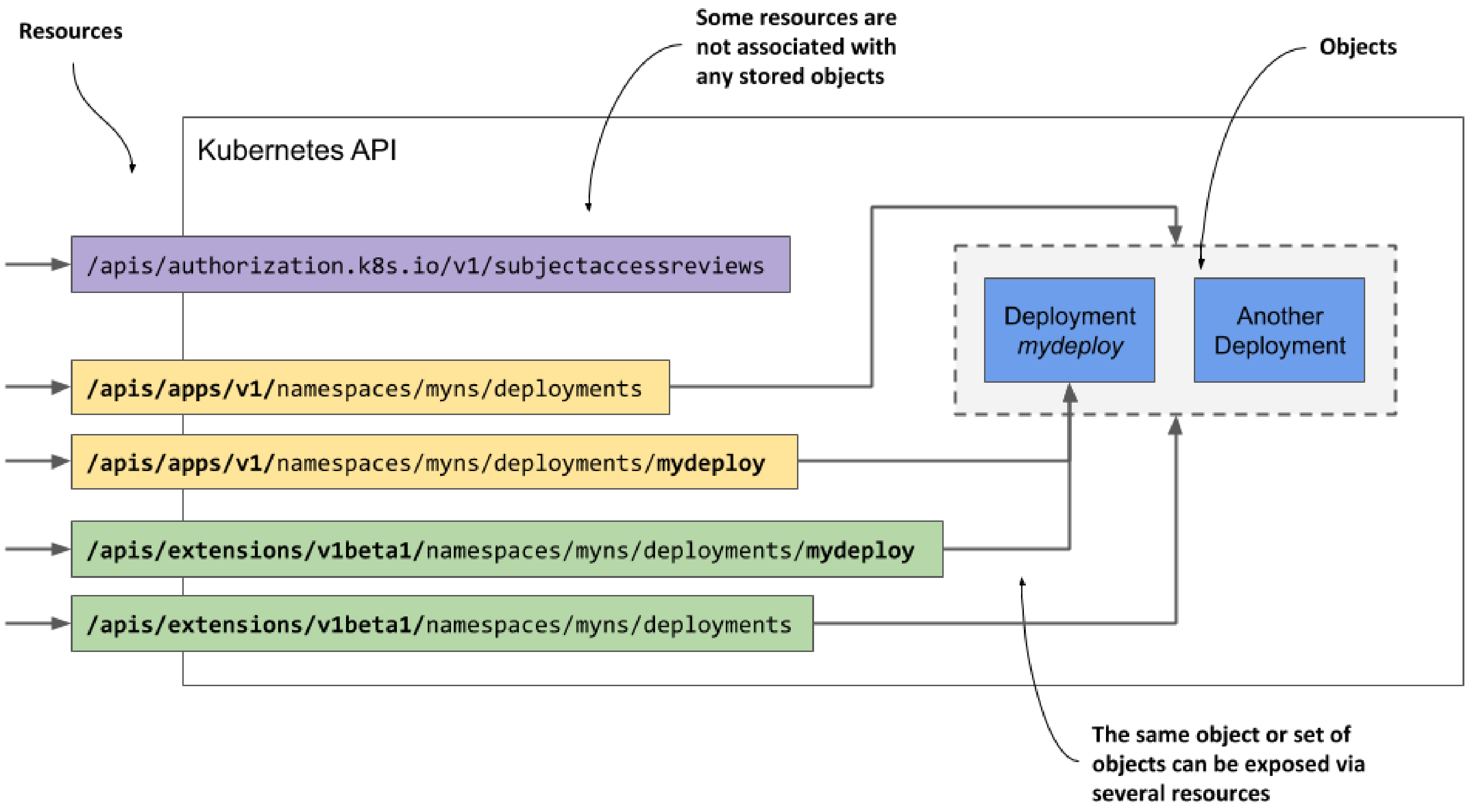
https://kubernetes.io/

Kubernetes high level Architecture

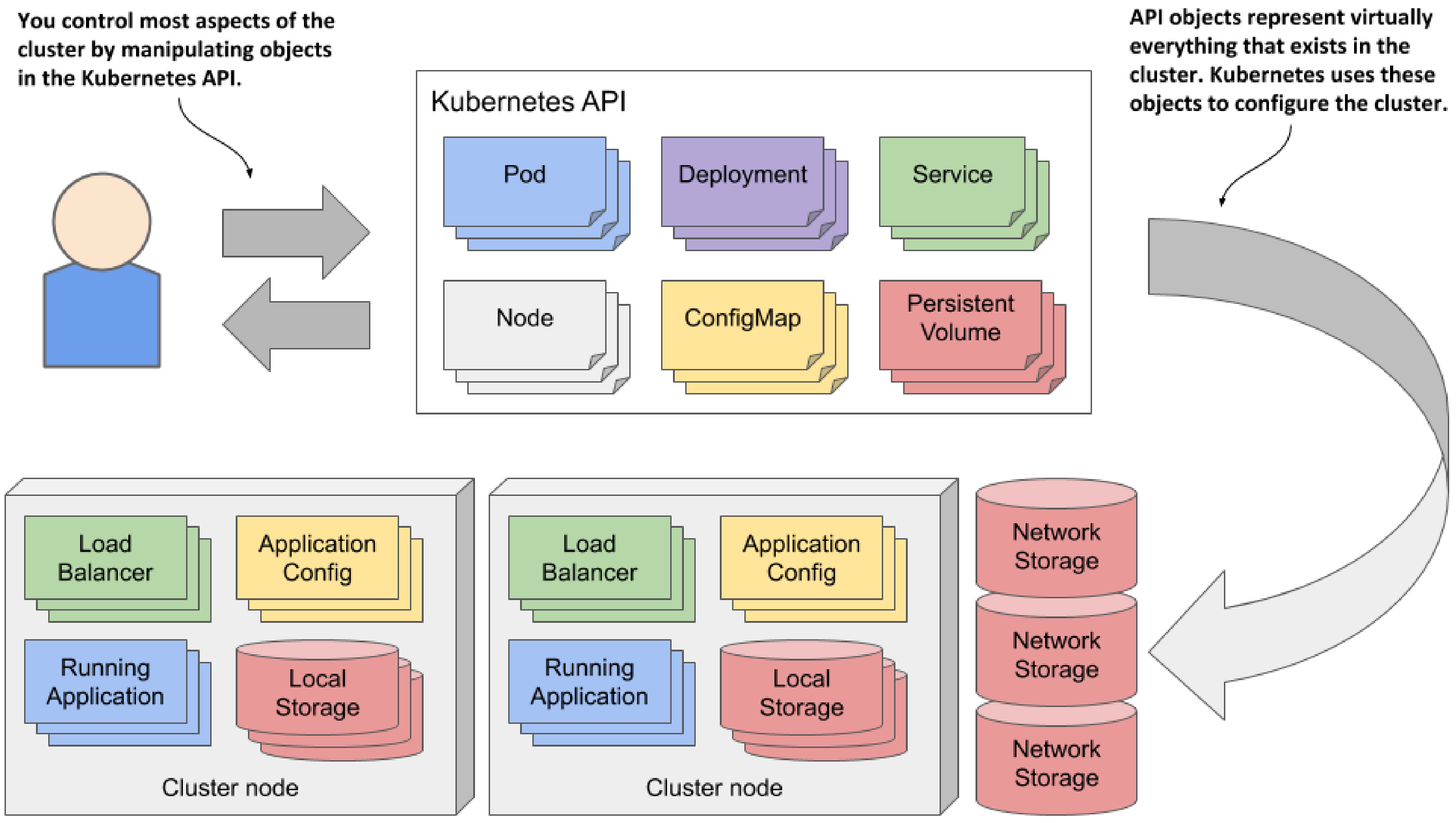
- **Highly distributed system**
  - Control loop pattern
  - Asynchronous / event-driven
  - Connect to API and listen to events
- Core of Kubernetes: **powerful API**
  - Every aspect is represented as API objects
  - Stores the serialized state of objects by writing them into etcd
  - Exposes REST API
  - Uses gRPC and protobuf for intra-cluster communication
  - Highly extensible
- Cloud provider specific integration due to **well defined interfaces**
  - CRI – Container Runtime Interface
  - CNI – Container Network Interface
  - CSI – Container Storage Interface



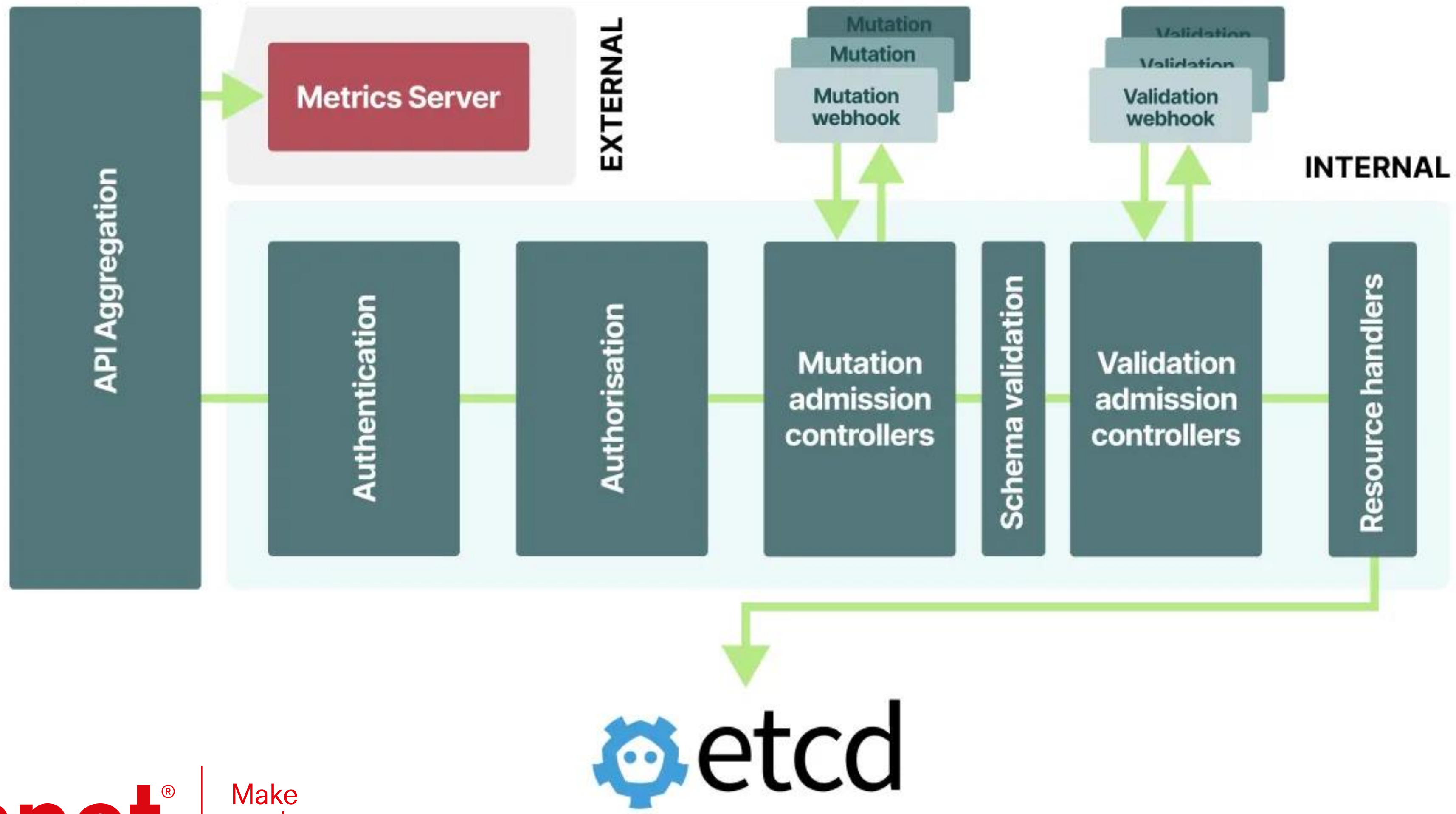
Kubernetes API: **resources and objects**



Kubernetes API: **objects represent infrastructure**



Kubernetes API: **processing pipeline**



https://medium.com/@danielepolencic/the-kubernetes-api-architecture-81da0ede0e34

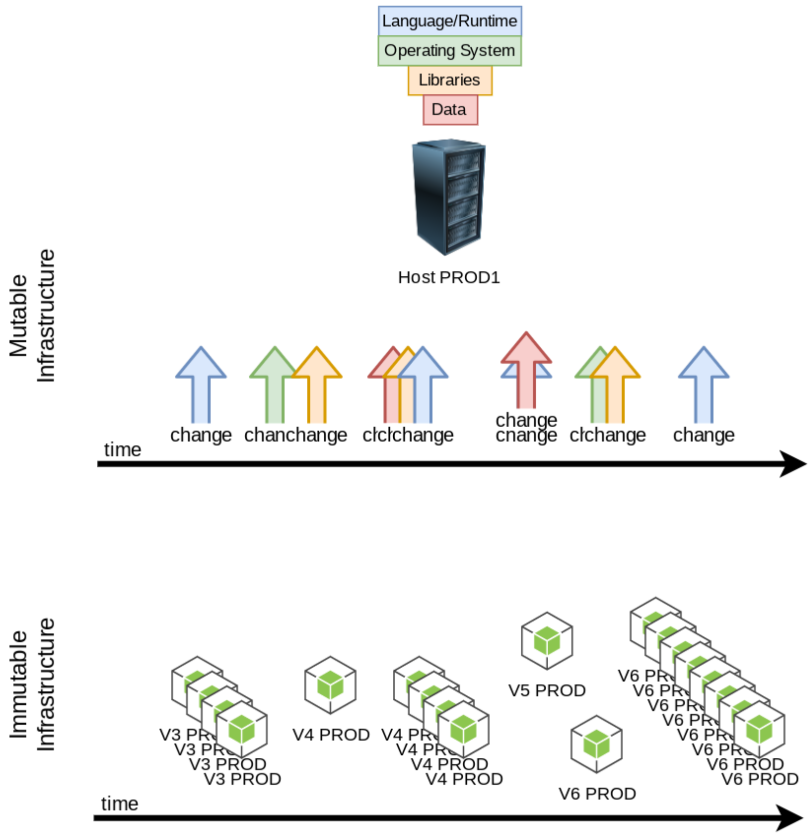
Underlying Concepts

claranet® Make modern happen.

K8s and the state: **immutable infrastructure**

- No in-place modifications possible any more
- CRUD -> **Update operations yield Delete and Create operations**
- Strict separation between stateless components and persistency layer
- Requires a different operational model: **cattle over pets**

- **Why?**
  - Reduced **complexity**
  - Reduced risk of unwanted **side effects**
  - Fully **versioned** infrastructure
  - **Well-known** server states
  - No **configuration drifts**
  - End to end **testing** of infrastructure stacks
  - Fewer deployment **failures**
  - Easy **rollbacks**



https://software.danielwatrous.com/immutable-infrastructure-production-release/

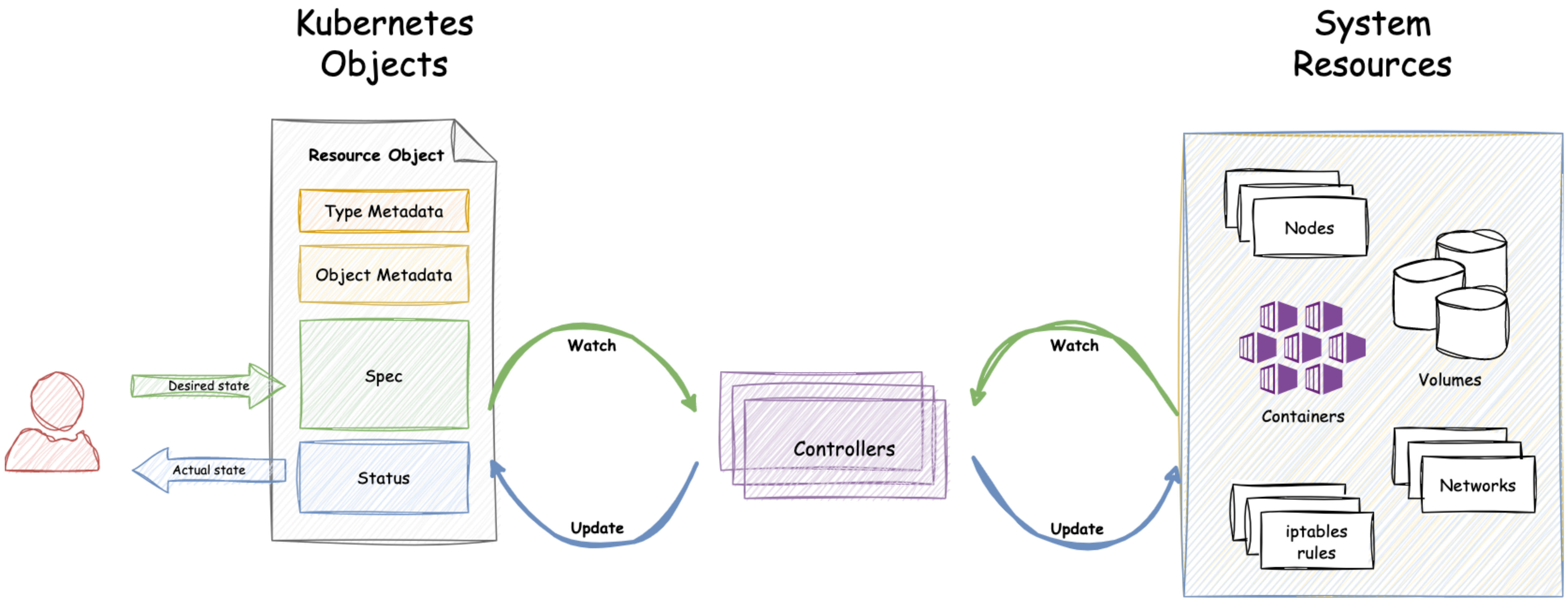
K8s and the state: **declarative code**

- Declarative API
  - Express desired state in YAML
  - Controllers taking care for the fulfillment
- Decouples user from implementation details
- Frees user from dealing with state
- Reduces complexity for the user

Imperative	Declarative
Specify the **how** to get the desired result by providing detailed instructions	Specify the **what** result is expected from the program
Direct the **control flow** of the program	Define the **expected result** without directing the program's control flow
The **developer** makes the major decisions about how the program works	The **compiler** makes the major decisions about how the program works
Code needs to deal with **current state**	Code only needs to state the **desired state**; controller implementation needs to deal with state
complex code	**simple** and clean code
It uses **mutable variables**, i.e., the values of variables can change during program execution.	It uses **immutable variables**, i.e., the values of variables cannot change

claranet® Make modern happen.

K8s and the state: **controller**



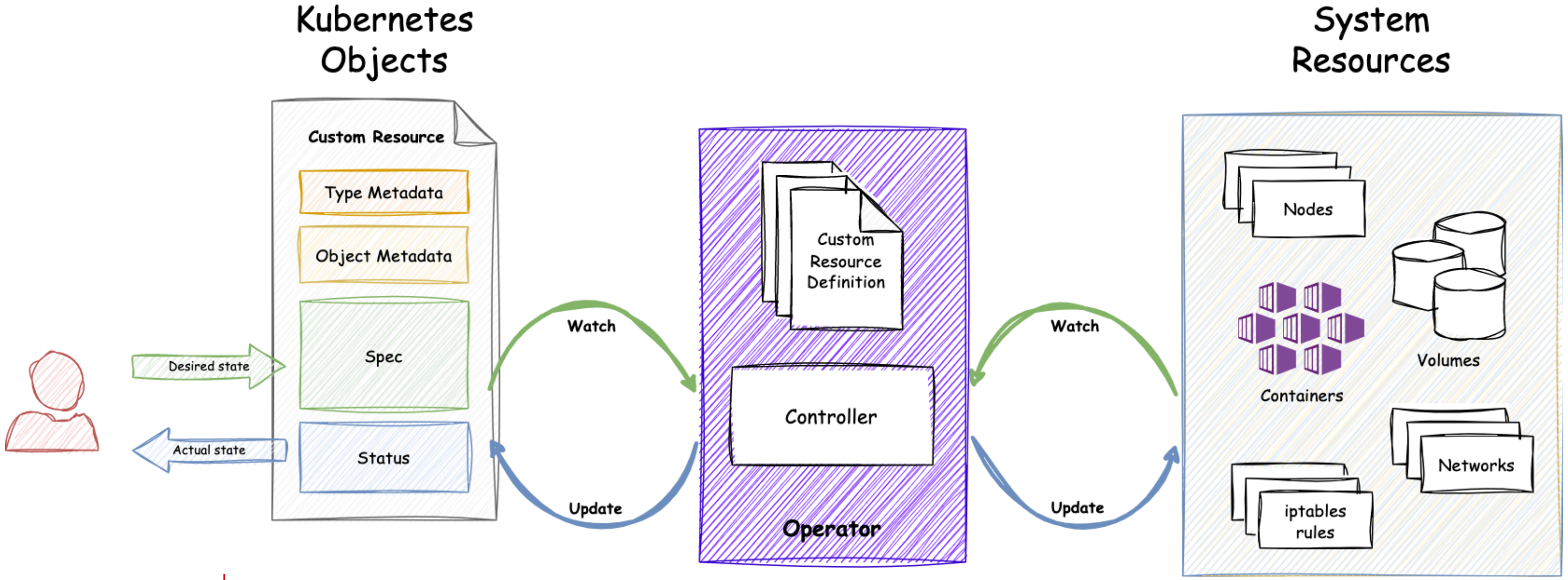
K8s and the state: **common controllers**

- Scheduler
  - watches for newly created Pods with no assigned node, and selects a node for them to run on
  - Factors taken into account for scheduling decisions include: individual and collective resource requirements, hardware/software/policy constraints, affinity and anti-affinity specifications, data locality, inter-workload interference, and deadlines
- Controllers
  - **Node controller:** Responsible for noticing and responding when nodes go down
  - **Job controller:** Watches for Job objects that represent one-off tasks, then creates Pods to run those tasks to completion.
  - **EndpointSlice controller:** Populates EndpointSlice objects (to provide a link between Services and Pods)
  - **ServiceAccount controller:** Create default ServiceAccounts for new namespaces.
- Cloud Controllers
  - **Node controller:** For checking the cloud provider to determine if a node has been deleted in the cloud after it stops responding
  - **Route controller:** For setting up routes in the underlying cloud infrastructure
  - **Service controller:** For creating, updating and deleting cloud provider load balancers
  - **Ingress controller:** For creating HTTP routing rules
  - **DNS Controller:** For creating A records in managed DNS zones to route traffic onto a certain domain

K8s as framework: **operator pattern**

- **Automate repetitive operational tasks**
- Extend API and introduce Custom Resources
- Implement custom controllers
- Listening for appropriate events
- Implement operational procedures as code

- Lifecycle management through CRDs versioning
- Examples
  - Let's Encrypt
  - Prometheus Operator
  - Argo CD
  - Postgres Operator
  - Istio

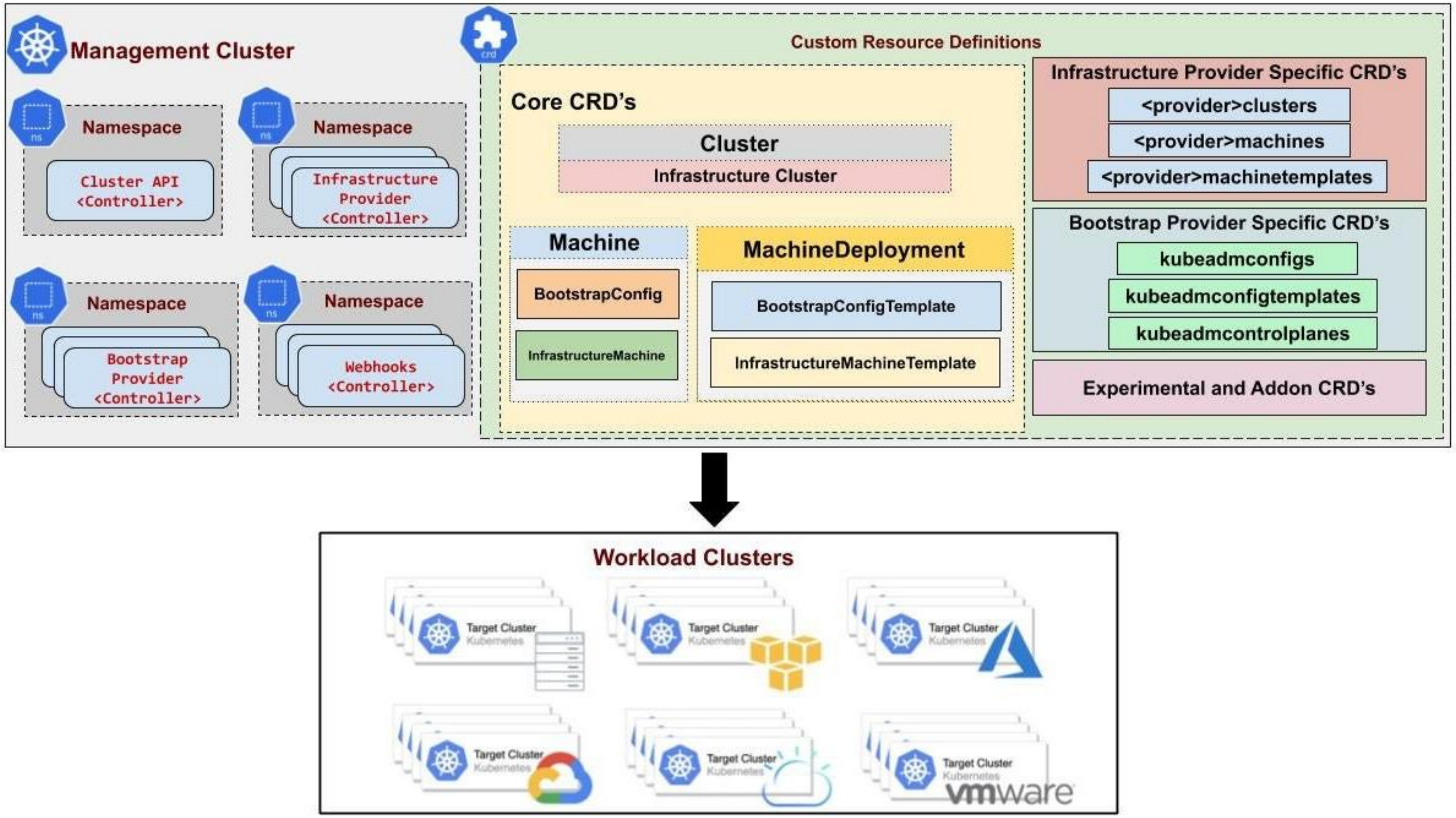


claranet® Make modern happen.

K8s as framework: **Cluster API**

Utilize the operator pattern to manage k8s cluster with the help of k8s

Cluster as a set of machines and underlying infrastructure represented as API objects



claranet® Make modern happen.

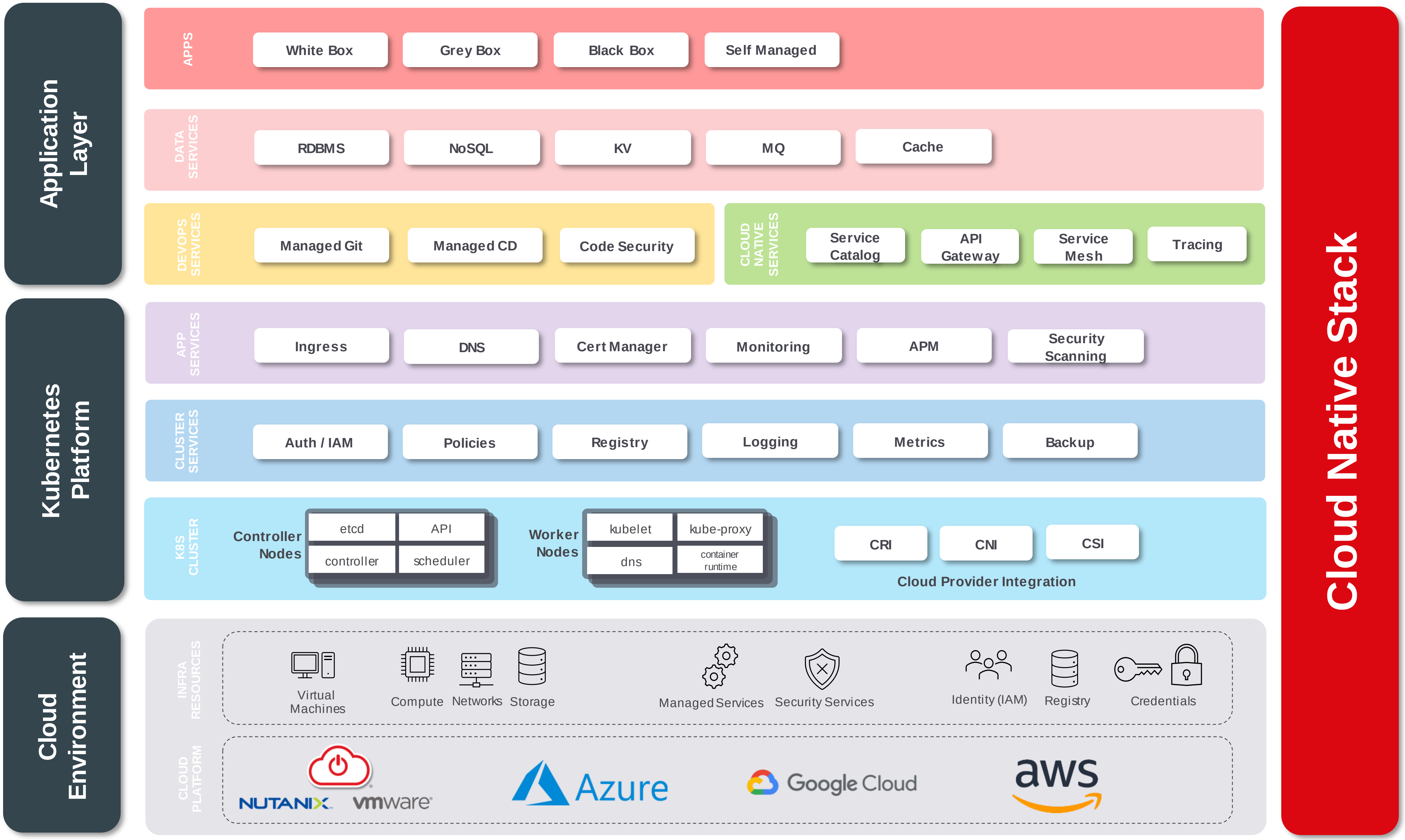
https://itnext.io/multi-cloud-and-multi-cluster-declarative-kubernetes-cluster-creation-and-management-with-cluster-6df8efdc2a89

Is it worth all the fuss?

"Most people are capable of building systems that are twice as complex as the systems they are capable of maintaining."

@casio_juarez

claranet® Make modern happen.



Application Layer
Kubernetes Platform
Cloud Environment
APPS
White Box
Grey Box
Black Box
Self Managed
DATA SERVICES
RDBMS
NoSQL
KV
MQ
Cache
DEVOPS SERVICES
Managed Git
Managed CD
Code Security
CLOUD NATIVE SERVICES
Service Catalog
API Gateway
Service Mesh
Tracing
APP SERVICES
Ingress
DNS
Cert Manager
Monitoring
APM
Security Scanning
CLUSTER SERVICES
Auth / IAM
Policies
Registry
Logging
Metrics
Backup
K8S CLUSTER
Controller Nodes
etcd
API
controller
scheduler
Worker Nodes
kubelet
kube-proxy
dns
container runtime
CRI
CNI
CSI
Cloud Provider Integration
INFRA RESOURCES
Virtual Machines
Compute
Networks
Storage
Managed Services
Security Services
Identity (IAM)
Registry
Credentials
CLOUD PLATFORM
NUTANIX
vmware
Azure
Google Cloud
aws
Cloud Native Stack

Why is Kubernetes that complex?

- **By intention:** we deliberately **introduce complexity as a strategy to deal with complexity**, but **complexity does not simply disappear**, it merely moves between layers and parties
- **Inherent nature of distributed systems**
- Platform incorporated more and more **generalised application functionalities**
- **Extensibility** is a two-edged sword
- **Flexibility** leaves a lot of knobs to turn
- **Plethora** of involved components
- **Lifecycle management**
  - Each cluster component is subject of lifecycle mgmt
  - Tight coupling of applications with runtime and dependencies puts developers in the drivers seat

Conclusions: complexity for the good

- **Harnessing immutable infrastructure paradigm reduces complexity**
- **Abstraction from infrastructure**
  - A developer never needs to login to a particular node any more
  - But in order to assess performance, infrastructure categories are still relevant
- **Allows to establish an explicit shared responsibility model between infrastructure-, platform-, and software engineering**
- **Degree of automation allows new ways of thinking**

Conclusions: strategic relevance

- Strategic relevance
  - **Kubernetes is more than only a container orchestration engine!**
  - **Universal control plane** for consuming data center services (compute, network, storage, IAM, resource control, scaling)
  - **A platform to build platforms**
  - The operating system of the cloud
  - Complexity shift from apps into the platform

- Key features:
  - Decouples workload from infrastructure
  - **Enables differentiation and specialisation** by introducing well-defined and robust interfaces
  - Portable across cloud vendors

- Recommended scenarios
  - Boosts application modernisation initiatives
  - Map microservices onto a fleet of compute nodes
  - Build custom PaaS platform
  - Operational framework for building SaaS products
  - Run AI stacks – batch training jobs
  - Unified security layer
  - Edge computing

Wanna join the **Cloud Native** movement?

Claranet, a place for talented people, partner-like customers, collaborative culture, personal growth, innovative technologies, vibrant international community

- Bachelor- / Master theses
- Cloud Native Engineer
- Cloud Native Consultant
- DevOps Engineer
- Kubernetes Engineer
- Site Reliability Engineer

Check out https://www.claranet.de/jobs

claranet®

Make modern happen.